Scalable Structure Discovery in Regression using Gaussian Processes

Hyunjik Kim HKIM@STATS.OX.AC.UK
Yee Whye Teh Y.W.TEH@STATS.OX.AC.UK
Department of Statistics, University of Oxford, 24-29 St Giles, Oxford OX1 3LB

Abstract

Automatic Bayesian Covariance Discovery (ABCD) in Lloyd et al. (2014) provides a framework for automating statistical modelling as well as exploratory data analysis for regression problems. However ABCD does not scale due to its $O(N^3)$ running time for the kernel search. This is undesirable not only because the average size of data sets is growing fast, but also because there is potentially more information in bigger data, implying a greater need for more expressive models that can discover finer structure. We propose Scalable Kernel Discovery (SKD), a scalable kernel search algorithm, to encompass big data within the boundaries of automated statistical modelling.

1. Introduction

Automated statistical modelling is an area of research in its early stages, yet it is becoming an increasingly important problem. As an increasing number of disciplines use statistical analyses and models to help achieve their goals, the demand for statisticians, machine learning researchers and data scientists is at an all time high. Automated systems for statistical modelling aim to serve as an assistant to help increase the efficiency of these human resources, if not as a best alternative where there is a shortage.

Duvenaud et al. (2013) take the first step of tackling the problem of structure discovery in nonparametric regression by fitting a Gaussian Process (GP) to the data, with an algorithm for automatically choosing a suitable parametric form of the kernel. This leads to high predictive performance that matches those with kernels hand-selected by GP experts (Rasmussen, 2006). There also exist other approaches that tackle this model selection problem by using a more flexible kernel (Wilson and Adams, 2013; Samo and Roberts, 2015). However the distinctive feature of Duvenaud et al. (2013) is that the resulting GPs are interpretable; the kernels are constructed in such a way that we can use them to describe patterns in the data, and thus can be used for automated exploratory data analysis. Lloyd et al. (2014) extend this to generate natural language analyses from these kernels, a procedure which they name Automatic Bayesian Covariance Discovery (ABCD). The Automatic Statistician[1] implements this to output a 10-15 page report when given data input.

However a limitation of ABCD is that it does not scale; due to the $O(N^3)$ time for inference in GPs, the analysis is constrained to small data sets, specialising on one dimensional time series data. This is a grave drawback in this era when data is getting bigger and more

1. See http://www.automaticstatistician.com/index/ for example analyses

high dimensional. Moreover it is clear that the importance of model selection increases with the size of the data set; we would like to select a more expressive model that adequately captures the information in the bigger data. This paper proposes a scalable extension to ABCD, to encompass big data within the boundaries of automated statistical modelling.

2. Automatic Bayesian Covariance Discovery (ABCD)

The Compositional Kernel Search (CKS) algorithm in Duvenaud et al. (2013) builds on the idea that the sum and product of two positive definite kernels are also positive definite. Starting off with a set $\mathcal{B}$ of base kernels defined on $\mathbb{R}\times\mathbb{R}$, the algorithm searches through the space of zero-mean GPs with kernels that can be expressed in terms of sums and products of these base kernels. The base set $\mathcal{B} = \{\text{SE,LIN,PER}\}$ is used, which correspond to the squared exponential, linear and periodic kernel respectively.[2] So candidate kernels form an open-ended space of GP models, allowing for an expressive model. A greedy search is employed to explore this space, with each kernel scored by the Bayesian Information Criterion (BIC) (Schwarz et al., 1978) after optimising the kernel hyperparameters by type II maximum likelihood(ML-II). See Appendix A for the algorithm in detail.

The resulting kernel can be simplified to be expressed as a sum of product of base kernels, which has the remarkable benefit of interpretation. In particular, note $f_1 \sim GP(0, k_1), f_2 \sim GP(0, k_2) \Rightarrow f_1 + f_2 \sim GP(0, k_1 + k_2)$. So a sum of products of kernels can be interpreted as sums of functions each with structure given by the product of kernels. Now each base kernel in a product modifies the model in a consistent way. For example, multiplication by SE converts global structure into local structure since $\text{SE}(x, x')$ decreases exponentially with $|x - x'|$, and multiplication by LIN is equivalent to multiplication of the modeled function by a linear function since $f(x) \sim GP(0, k) \Rightarrow xf(x) \sim GP(0, k \times LIN)$.[3] Lloyd et al. (2014) uses this observation for ABCD, giving a natural language description of the resulting function modeled by the composite kernel. In summary ABCD consists of two algorithms: the compositional kernel search CKS, and the natural language translation of the kernel into a piece of exploratory data analysis.

3. Scaling up ABCD

ABCD in Section 2 provides a framework for a natural extension to big data settings, in that we only need to be able to scale up the compositional kernel search algorithm, then the natural language description of models can be directly applied.

The difficulty of the extension of the compositional kernel search to big data settings lies in the $O(N^3)$ time for evaluation of the GP marginal likelihood and its gradients with respect to the kernel hyperparameters. It is tempting to use as a proxy for the exact likelihood either an approximate marginal likelihood or the exact marginal likelihood of an approximate model. However we will need practically meaningful guarantees that relate these approximate quantities to the exact marginal likelihood of the full GP model, in order for the chosen kernel to faithfully reflect the actual structure in the data. These guarantees on known approximate GP marginal likelihoods are often difficult to achieve. Instead we

2. The exact form of these base kernels are given in Appendix B
3. See Lloyd et al. (2014) for detailed interpretations for different base kernels

provide a lower and upper bound to sandwich the exact marginal likelihood, and we use this interval for model selection. To do so we give a brief overview of the relevant work on low rank kernel approximations used for scaling up GPs, and we later outline how they can be applied to obtain cheap lower and upper bounds.

3.1. Random Fourier Features

Random Fourier Features (RFF) (a.k.a.Random Kitchen Sinks) was introduced by Rahimi and Recht (2007), which uses Bochner's theorem (Rudin, 1964) to give an unbiased low-rank approximation to the Gram matrix $K = \mathbb{E}[\Phi^\top \Phi]$ with $\Phi \in \mathbb{R}^{m \times N}$ (see Appendix C for details). A bigger m lowers the variance of the estimate. Using this approximation, one can compute determinants and inverses in $O(Nm^2)$ time. In the context of kernel composition in Section 2, RFFs have the nice property that samples from the spectral density of the sum or product of kernels can easily be obtained as sums or mixtures of samples of the individual kernels (see Appendices C and D). We use this later to give a memory-efficient upper bound on the exact marginal likelihood.

3.2. Nyström Methods and Sparse Gaussian Processes

The Nyström Method (Williams and Seeger, 2001; Drineas and Mahoney, 2005) selects a set of m inducing points in the input space $\mathbb{R}^D$ that attempt to explain all the covariance in the Gram matrix of the kernel: the kernel is evaluated for each pair of inducing points and also between the inducing points and the data, giving matrices $K_{mm}, K_{mn} = K_{nm}^\top$. This is used to create the Nyström approximation $\hat{K} = K_{nm}(K_{mm})^\dagger K_{mn}$ of K. Applying the Cholesky decomposition to K_{mm}, we see that the approximation admits the form $\Phi^\top \Phi$ and so allow efficient computations as for RFF. We later use the Nyström approximation to give another upper bound on the exact marginal likelihood.

The Nyström approximation arises naturally in the sparse GP literature, where certain distributions are approximated by simpler ones involving $\boldsymbol{f_m}$, the GP evaluated at the m inducing points: the DTC approximation of Seeger et al. (2003) defines a model that gives the marginal likelihood $q(y) = \mathcal{N}(y|0, \hat{K} + \sigma^2 I)$, whereas the FIC approximation of Snelson and Ghahramani (2005) gives $q(y) = \mathcal{N}(y|0, \hat{K} + \mathrm{diag}(K - \hat{K}) + \sigma^2 I)$, correcting the Nyström approximation along the diagonals. Quinonero-Candela and Rasmussen (2005) further improves this by introducing the PIC approximation, where the Nyström approximation is corrected on block diagonals with blocks typically of size $m \times m$. Note however for FIC and PIC that the approximation is no longer low rank, but matrix inversion can still be computed efficiently by Woodbury's Lemma (see Appendix F).

The variational inducing points method (VAR) introduced by Titsias (2009) is rather different to DTC/FIC/PIC in that it gives the following variational lower bound on the exact log marginal likelihood (see paper for derivation):

$$\log[\mathcal{N}(y|0, \hat{K} + \sigma^2 I)] - \frac{1}{2\sigma^2} \mathrm{Tr}(K - \hat{K}) \tag{1}$$

This lower bound is optimised with respect to the inducing points and the kernel hyperparameters, which is shown in the paper to successfully yield tight lower bounds in $O(Nm^2)$ time for reasonable values of m. Another useful property of VAR is that the lower bound

can only increase as the set of inducing points grows (Titsias, 2009; Matthews et al., 2016). Bauer et al. (2016) also points out that VAR always improves with extra computation, and that it successfully recovers the true posterior GP in most cases, contrary to other sparse GP methods. Hence this is what we use in the scalable structure discovery to obtain a lower bound on the marginal likelihood and optimise the hyperparameters. Also note that contrary to DTC/FIC/PIC, Equation (1) cannot be seen as the log marginal likelihood with a plug-in estimate for the Gram matrix.

3.3. A Cheap Upper Bound on the Marginal Likelihood

Fixing the hyperparameters to be those tuned by VAR, we seek a cheap upper bound to the exact marginal likelihood. Upper bounds and lower bounds are qualitatively different, and in general it is more difficult to obtain an upper bound than a lower bound for the following reason: first note that the marginal likelihood is the integral of the likelihood with respect to the prior density of parameters. Hence to obtain a lower bound it suffices to exhibit regions in the parameter space giving high likelihood. On the other hand, to obtain an upper bound one must demonstrate the absence or lack of likelihood mass outside a certain region. There has been some work on the subject (Beal, 2003; Ji et al., 2010), but to the best of our knowledge there has not been any work on cheap upper bounds to the marginal likelihood affordable in large N settings. So finding an upper bound from the perspective of the marginal likelihood can be difficult. Instead, we exploit the fact that the GP marginal likelihood has an analytic form, and treat it as a function of K.

The GP marginal likelihood is composed of two terms and a constant:

$$\log p(y) = -\frac{1}{2}\log\det(K+\sigma^2 I) - \frac{1}{2}y^\top (K+\sigma^2 I)^{-1} y - \frac{N}{2}\log(2\pi) \tag{2}$$

We give separate upper bounds on the negative log determinant (NLD) term and the negative inner product (NIP) term. For NLD, we give two candidate upper bounds. Firstly, Bardenet and Titsias (2015) prove that

$$-\frac{1}{2}\log\det(K+\sigma^2 I) \leq -\frac{1}{2}\log\det(\hat{K}+\sigma^2 I) \tag{3}$$

a consequence of $K-\hat{K}$ being positive semi-definite. Hence the Nyström approximation plugged into the NLD term serves as an upper bound. Note this can be computed in $O(Nm^2)$ time by Sylvester's Determinant Theorem (Appendix F).

Alternatively, note that the function $f(X) = -\log\det(X)$ is convex on the set of positive definite matrices (Boyd and Vandenberghe, 2004). Hence by Jensen's inequality we have, for $\Phi^\top\Phi$ an unbiased estimate of K:

$$-\frac{1}{2}\log\det(K+\sigma^2 I) = f(K+\sigma^2 I) = f(\mathbb{E}[\Phi^\top\Phi+\sigma^2 I]) \leq \mathbb{E}[f(\Phi^\top\Phi+\sigma^2 I)] \tag{4}$$

Hence $-\frac{1}{2}\log\det(\Phi^\top\Phi+\sigma^2 I)$ is a stochastic upper bound to NLD that can be calculated in $O(Nm^2)$. An example of such an unbiased estimator Φ is given by RFF. We compare these two upper bounds to NLD in Section 4.

As for NIP, we point out that $\lambda y^\top (K+\sigma^2 I)^{-1} y$ is the optimal value of the objective function in kernel ridge regression $\min_{f\in\mathcal{H}} \sum_{i=1}^N (y_i - f(x_i))^2 + \lambda\|f\|_{\mathcal{H}}^2$, where $\mathcal{H}$ is the

Algorithm 1: Scalable Kernel Discovery algorithm

Input: data $x_1, \ldots, x_n \in \mathbb{R}^D, y_1, \ldots, y_n \in \mathbb{R}$, base kernel set $\mathcal{B}$, depth d, maximum number of inducing points m, kernel buffer size S

Output: k, the resulting kernel

For each base kernel on each dimension, obtain lower and upper bounds to BIC (BIC interval), set k to be the kernel with highest lower bound, and add k to kernel buffer $\mathcal{K}$.

$\mathcal{C} \leftarrow \emptyset$

for *depth=1:d* **do**

  From $\mathcal{C}$, add to $\mathcal{K}$ all kernels whose intervals overlap with k if there are fewer than S of them, else add the kernels with top S lower bounds.

  for $k' \in \mathcal{K}$ **do**

    Add following kernels to $\mathcal{C}$ and obtain their BIC intervals:

      (1) All kernels of form $k' + B$ where B is any base kernel on any dimension

      (2) All kernels of form $k' \times B$ where B is any base kernel on any dimension

      (3) All kernels where a base kernel in k' is replaced by another base kernel

  end

  if *exists kernel* $k^* \in \mathcal{C}$ *with higher lower bound than* k **then**

    $k \leftarrow k^*$

  end

end

Reproducing Kernel Hilbert space associated with k. The dual problem, whose objective function has the same optimal value, is $\max_{\alpha \in \mathbb{R}^N} -\lambda[\alpha^\top (K + \sigma^2 I)\alpha - 2\alpha^\top y]$. Noting that $\max_\alpha -g(\alpha) = -\min_\alpha g(\alpha)$, replacing λ with σ^2 and multiplying by a suitable constant we have:

$$-\frac{1}{2} y^\top (K + \sigma^2 I)^{-1} y = \min_{\alpha \in \mathbb{R}^N} \frac{1}{2} \alpha^\top (K + \sigma^2 I)\alpha - \alpha^\top y \tag{5}$$

Hence $\frac{1}{2}\alpha^\top (K + \sigma^2 I)\alpha - \alpha^\top y$ is an upper bound for NIP $\forall \alpha \in \mathbb{R}^N$. Note that this is also in the form of an objective for conjugate gradients(CG) (Shewchuk, 1994), so the optimal value is at $\hat{\alpha} = (K + \sigma^2 I)^{-1} y$. We can approach the optimum for a tighter bound by using CG or preconditioned CG (PCG) for $O(m)$ iterations to get a reasonable approximation to $\hat{\alpha}$. Each iteration of CG and the computation of the upper bound takes $O(N^2)$ time, but PCG is very fast for large data sets and FIC/PIC give fastest convergence in general (Cutajar et al., 2016). Also note that we only need to compute the upper bound once, whereas one must evaluate the lower bound and its gradients multiple times for the hyperparameter optimisation. We later confirm in Section 4 that the upper bound is fast to compute relative to the lower bound optimisation. We also provide results to show the effectiveness of this upper bound for various kernel approximations.

3.4. SKD: Scalable Kernel Discovery using the Lower and Upper Bound

Given a kernel and a value of m, we can compute the lower and upper bounds as above to obtain an interval for the exact GP marginal likelihood and hence the BIC of the kernel with its hyperparameters optimised by VAR. These hyperparameters may of course not

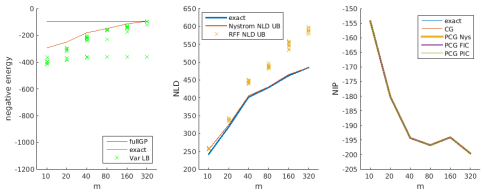
be the global maximisers of the exact GP marginal likelihood, but as in ABCD we may optimise the marginal likelihood with multiple sets of random starting values to find the local optimum closest to the global optimum.

Note that we can guarantee that the lower bound increases with larger m, but cannot guarantee that the upper bound decreases. In fact, the upper bound is likely to increase as well, since with larger m it is likely that one can find hyperparameters that give a higher exact marginal likelihood, hence a higher upper bound. We verify this in later experiments. Hence for kernel evaluation, it would be sensible to use the largest possible value of m that one can afford, so that the exact marginal likelihood with hyperparameters optimised by VAR is as close as possible to the exact marginal likelihood with optimal hyperparameters.

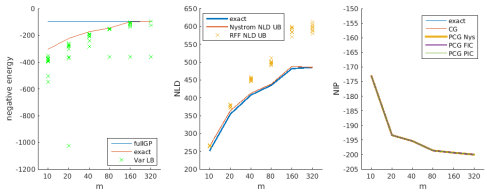
With these intervals for each kernel, we can perform a semi-greedy kernel search whereby we expand the kernel tree on the top S intervals of the current depth. A summary of the Scalable Kernel Discovery algorithm is given in Algorithm 1. Further details on the optimisation and initialisation are given in Appendix G.

4. Experiments

We present results for experiments showing the bounds we obtain for two small time series and a multidimensional regression data set, for which ABCD is feasible. The first is the annual solar irradiance data from 1610 to 2011, with 402 observations (Lean et al., 1995) where we use the kernel (SE+SE)×PER. The second is the time series Mauna Loa CO2 data [4] with 689 observations with kernel SE×LIN + SE×PER. The multidimensional data set is the concrete compressive strength data set with 1030 observations and 8 covariates.[5] with kernel LIN4+SE1 × SE7+SE1 × SE2 × SE4 + SE2 × SE4 × SE8 + SE2 × SE4 × SE7 × SE8 × LIN4. All these kernels have been found by CKS. All observations and covariates have been normalised to have mean 0 and variance 1.



(a) Solar: fix inducing points



(b) Solar: learn inducing points

Figure 1: (a) Left: negative energy (lgo marginal likelihood + log prior) for fullGP with optimised hyperparameters, optimised VAR lower bound for each of the 10 random initialisations per m, and the exact negative energy for the best hyperparameters out of the 10. Middle: exact NLD and upper bounds. Right: exact NIP and upper bounds after m iterations of CG/PCG. (b) Same as Figure 1(a), except learning inducing points for the VAR lower bound optimisation and using them for subsequent computations.

4. Data can be found at ftp://ftp.cmdl.noaa.gov/ccg/co2/trends/co2_mm_mlo.txt
5. Data can be found at https://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength

From the left of Figures 1(*a*), 4(*a*) and 5(*a*) (the latter two can be found in Appendix K) we see that VAR gives a lower bound for the optimal marginal likelihood that improves with increasing m. From the variance of the different random initialisations, we see that having around 10 initialisations seems sufficient. The best lower bound is tight relative to the exact marginal likelihoods at the hyperparameters optimised by VAR. From the middle plots, we observe that the Nyström approximation gives a very tight upper bound on the NLD term, and does indeed increase with m. RFF gives upper bounds that are not as tight, especially for larger values of m. From the right plots, we can see that PCG with any of the three preconditioners (Nyström, FIC, PIC) give very tight upper bounds to the NIP term, whereas CG may require more iterations to get tight, for example in Figures 4(*a*) and 4(*b*) (latter two can be found in Appendix K).

Comparing Figures 1(*a*), 4(*a*) and 5(*a*) against Figures 1(*b*), 4(*b*) and 5(*b*), learning inducing points does not lead to a vast improvement in the VAR lower bound. In fact the differences are not very significant, and sometimes learning inducing points can get the lower bound stuck in a bad local minimum, as indicated by the high variance of lower bounds in the latter three figures. Moreover the differences in computational time is significant as we can see in Table 2 of Appendix I. Hence the computation-accuracy trade-off is best when fixing the inducing points.

Fixing the inducing points, we also compare times for the different computations in Table 2. The gains from using the variational lower bound instead of the full GP is clear, especially for the larger data sets, and we also confirm that it is indeed the optimisation of the LB that is the bottleneck in terms of computational cost. We also see that the NIP upper bound computation times are similarly fast for all m, thus convergence of PCG with the PIC preconditioner is happening in only a few iterations.

Table 1: Kernels found by CKS and SKD for different values of m on the three datasets.

	Solar	**Mauna**	**Concrete**
Best	SE × PER	SE × PER	SE8
CKS	SE × PER	SE × PER	SE8
SKD, m=10	SE × LIN	SE × PER	SE8
m=20	SE × PER	SE × PER	PER8
m=40	SE × PER	SE + PER	PER8
m=80	SE × PER	SE × PER	PER8
m=160	SE × PER	SE × PER	PER8
m=320	SE × PER	SE + PER	PER8

We also compare the kernels chosen by CKS and by SKD up to depth 2 for solar and Mauna, and depth 1 for concrete. The results are summarised in Table 1, with plots in Figures 2, 6 and 7 (the latter two can be found in Appendix K). Note that linear kernels have Gram matrices with rank 1, so the marginal likelihood does not improve much with greater m that gives higher rank approximations, hence the straight lines for LIN in the plots. We see that for solar, our method successfully finds the best kernel for $m = 20$. For Mauna, we find that our method alternates between SE × PER and SE+PER. The kernel with highest negative energy is SE × PER, but we observe that the lower bound for SE+PER for $m = 320$ is higher than the best GP negative energy for SE × PER out of

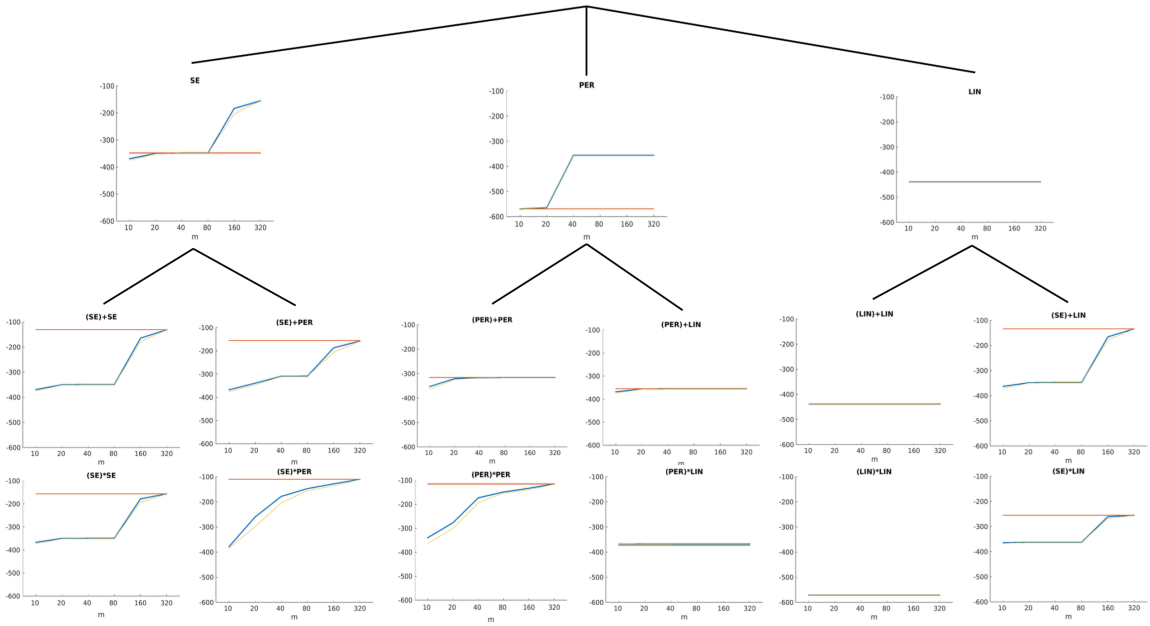


Figure 2: Kernel Tree search on solar data up to depth 2. The red, blue and yellow lines correspond to the best GP negative energy out of 10 random initialisations, the best lower bound out of 10 random initialisations, and the upper bound corresponding to the best lower bound, obtained by m iterations of PCG with PIC preconditioner.

10 random initialisations. Hence had the optimisation for the full GP been more thorough, we would have recovered the best kernel. For concrete, our methods finds PER8, although SE8 is the best kernel. However from Figure 2 we see that these two kernels show similar negative energy and are the top two kernels. So we see that our method successfully finds good kernels for low values of m.

5. Conclusion and Future Work

We have introduced SKD, a scalable kernel discovery algorithm that extends CKS and hence ABCD to bigger data sets. We have confirmed that SKD works well for small data sets where ABCD is feasible. Next, we should test whether SKD finds suitable kernels for medium sized data sets where ABCD is infeasible, but a single evaluation of GP marginal likelihood is just about feasible. Then we should go on to test SKD on large data sets with tens of thousands of data points. We should also look into using grid integration for evaluation of kernels, which would give a more accurate estimate of the model evidence than BIC. Note Laplace approximation adds on an expensive Hessian term, for which it is unclear how one can obtain lower and upper bounds.

References

Rémy Bardenet and Michalis Titsias. Inference for determinantal point processes without spectral knowledge. *NIPS*, 2015.

Matthias Stephan Bauer, Mark van der Wilk, and Carl Edward Rasmussen. Understanding probabilistic sparse gaussian process approximations. *NIPS*, 2016.

Matthew James Beal. *Variational algorithms for approximate Bayesian inference*. PhD thesis, 2003.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

Kurt Cutajar, Michael A. Osborne, John P. Cunningham, and Maurizio Filippone. Preconditioning kernel matrices. *ICML*, 2016.

Petros Drineas and Michael Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. *JMLR*, 6:2153–2175, 2005.

David Duvenaud, James Lloyd, Roger Grosse, Joshua Tenenbaum, and Ghahramani Zoubin. Structure discovery in nonparametric regression through compositional kernel search. In *ICML*, 2013.

Chunlin Ji, Haige Shen, and Mike West. Bounded approximations for marginal likelihoods. 2010.

Judith Lean, Juerg Beer, and Raymond S Bradley. Reconstruction of solar irradiance since 1610: Implications for climate cbange. *Geophysical Research Letters*, 22(23), 1995.

James Robert Lloyd, David Duvenaud, Roger Grosse, Joshua Tenenbaum, and Zoubin Ghahramani. Automatic construction and natural-language description of nonparametric regression models. In *AAAI*, 2014.

Alexander G de G Matthews, James Hensman, Richard E Turner, and Zoubin Ghahramani. On sparse variational methods and the kullback-leibler divergence between stochastic processes. *AISTATS*, 2016.

Joaquin Quinonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *The Journal of Machine Learning Research*, 6: 1939–1959, 2005.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *NIPS*, 2007.

Carl Edward Rasmussen. Gaussian processes for machine learning. 2006.

Walter Rudin. Fourier analysis on groups. *AMS*, 1964.

Yves-Laurent Kom Samo and Stephen Roberts. Generalized spectral kernels. *arXiv preprint arXiv:1506.02236*, 2015.

Gideon Schwarz et al. Estimating the dimension of a model. *The Annals of Statistics*, 6(2): 461–464, 1978.

Matthias Seeger, Christopher Williams, and Neil Lawrence. Fast forward selection to speed up sparse gaussian process regression. In *AISTATS*, 2003.

Jonathan Richard Shewchuk. An introduction to the conjugate gradient method without the agonizing pain, 1994.

Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In *NIPS*, 2005.

Arno Solin and Simo Särkkä. Explicit link between periodic covariance functions and state space models. 2014.

Michalis K Titsias. Variational learning of inducing variables in sparse gaussian processes. In *AISTATS*, 2009.

Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *NIPS*, 2001.

Andrew Gordon Wilson and Ryan Prescott Adams. Gaussian process kernels for pattern discovery and extrapolation. *arXiv preprint arXiv:1302.4245*, 2013.

Appendix A. Compositional Kernel Search Algorithm

Algorithm 2: Compositional Kernel Search Algorithm

Input: data $x_1, \ldots, x_n \in \mathbb{R}^D, y_1, \ldots, y_n \in \mathbb{R}$,base kernel set $\mathcal{B}$
Output: k, the resulting kernel
For each base kernel on each dimension, fit GP to data (i.e. optimise hyperparams by ML-II) and set k to be kernel with smallest BIC.
for *depth=1:T (either fix T or repeat until BIC no longer decreases)* **do**
  Fit GP to following kernels and set k to be the one with lowest BIC:
  (1) All kernels of form $k + B$ where B is any base kernel on any dimension
  (2) All kernels of form $k \times B$ where B is any base kernel on any dimension
  (3) All kernels where a base kernel in k is replaced by another base kernel
end

Appendix B. Base Kernels

$$\begin{aligned}
\mathrm{LIN}(x, x') &= \sigma^2 x x' \\
\mathrm{SE}(x, x') &= \sigma^2 \exp\Big(-\frac{(x - x')^2}{2l^2} \Big) \\
\mathrm{PER}(x, x') &= \sigma^2 \exp\Big(-\frac{2\sin^2(\pi(x - x')/p)}{l^2} \Big)
\end{aligned}$$

Appendix C. Random Fourier Features for Sums and Products of Kernels

Theorem 1 (Bochner's Theorem (Rudin, 1964)) *A stationary kernel k(d) is positive definite if and only if k(d) is the Fourier transform of a non-negative measure.*

For RFF the kernel can be approximated by the inner product of random features given by samples from its spectral density, in a Monte Carlo approximation, as follows:

$$\begin{aligned}
k(x - y) = \int_{\mathbb{R}^D} e^{iv^\top (x-y)} d\mathbb{P}(v) \propto \int_{\mathbb{R}^D} p(v) e^{iv^\top (x-y)} dv &= \mathbb{E}_{p(v)}[e^{iv^\top x}(e^{iv^\top y})^*] \\
&= \mathbb{E}_{p(v)}[Re(e^{iv^\top x}(e^{iv^\top y})^*)] \\
&\approx \frac{1}{m} \sum_{k=1}^{m} Re(e^{iv_k{}^\top x}(e^{iv_k{}^\top y})^*) \\
&= \mathbb{E}_b \phi(x)^\top \phi(y)
\end{aligned}$$

where $\phi(x) = \sqrt{\frac{2}{m}}(cos(v_1{}^\top x + b_1), \ldots, cos(v_m{}^\top x + b_m))$ with spectral frequencies v_k iid samples from $p(v)$ and b_k iid samples from $U[0, 2\pi]$.
Let k_1, k_2 be two stationary kernels, with respective spectral densities p_1, p_2 so that $k_1(d) = a_1 \hat{p_1}(d), k_2(d) = a_2 \hat{p_2}(d)$, where $\hat{p}(d) := \int_{\mathbb{R}^D} p(v) e^{iv^\top d} dv$. We use this convention as

the Fourier transform. Note $a_i = k_i(0)$.

$$(k_1 + k_2)(d) = a_1 \int p_1(v) e^{iv^\top d} dv + a_2 \int p_2(v) e^{iv^\top d} dv = (a_1 + a_2)\hat{p_+}(d)$$

where $p_+(v) = \frac{a_1}{a_1+a_2} p_1(v) + \frac{a_2}{a_1+a_2} p_2(v)$, a mixture of p_1 and p_2. So to generate RFF for $k_1 + k_2$, generate $v \sim p_+$ by generating $v \sim p_1$ w.p. $\frac{a_1}{a_1+a_2}$ and $v \sim p_2$ w.p. $\frac{a_2}{a_1+a_2}$
Now for the product, suppose

$$(k_1 \cdot k_2)(d) = a_1 a_2 \hat{p_1}(d)\hat{p_2}(d) = a_1 a_2 \hat{p_*}(d)$$

Then $p_*(d)$ is the inverse fourier transform of $\hat{p_1}\hat{p_2}$, which is the convolution $p_1 * p_2(d) := \int_{\mathbb{R}^D} p_1(z)p_2(d-z)dz$. So to generate RFF for $k_1 \cdot k_2$, generate $v \sim p_*$ by generating $v_1 \sim p_1, v_2 \sim p_2$ and setting $v = v_1 + v_2$.
This is not applicable for non-stationary kernels, such as the linear kernel. We show how this is dealt with in Appendix D.

Appendix D. Random Features for Sums and Products of Kernels

Suppose ϕ_1, ϕ_2 are random features such that
$k_1(x, x') = \phi_1(x)^\top \phi_1(x'), \phi_2(x)^\top \phi_2(x'), \phi_i : \mathbb{R}^D \to \mathbb{R}^m$.
It is straightforward to verify that

$$(k_1 + k_2)(x, x') = \phi_+(x)^\top \phi_+(x') \text{ where } \phi_+(\cdot) = (\phi_1(\cdot)^\top, \phi_2(\cdot)^\top)^\top$$
$$(k_1 \cdot k_2)(x, x') = \phi_*(x)^\top \phi_*(x') \text{ where } \phi_*(\cdot) = \phi_1(\cdot) \otimes \phi_2(\cdot)$$

However we do not want the number of features to grow as we add or multiply kernels, since it will grow exponentially. We want to keep it to be m features. So we subsample m entries from ϕ_+ (or ϕ_*) and scale by factor $\sqrt{2}$ ($\sqrt{m}$ for ϕ_*), which will still give us unbiased estimates of the kernel provided that each term of the inner product $\phi_+(x)^\top \phi_+(x')$ (or $\phi_*(x)^\top \phi_*(x')$) is an unbiased estimate of $(k_1 + k_2)(x, x')$(or $(k_1 \cdot k_2)(x, x')$).
This is how we generate random features for linear kernels combined with other stationary kernels, using the features $\phi(x) = \frac{\sigma}{\sqrt{m}}(x, \ldots, x)^\top$.

Appendix E. Spectral Density for PER

From Solin and Särkkä (2014), we have that the spectral density of the PER kernel is:

$$\sum_{n=-\infty}^{\infty} \frac{I_n(l^{-2})}{\exp(l^{-2})} \delta\left(v - \frac{2\pi n}{p}\right)$$

where I is the modified Bessel function of the first kind.

Appendix F. Matrix Identities

Lemma 2 (Woodbury's Matrix Inversion Lemma)
$(A + UBV)^{-1} = A^{-1} - A^{-1}U(B^{-1} + VA^{-1}U)^{-1}VA^{-1}$

Lemma 3 (Sylvester's Determinant Theorem)
$\det(I + AB) = \det(I + BA) \forall A \in \mathbb{R}^{m \times n} \forall B \in \mathbb{R}^{n \times m}$

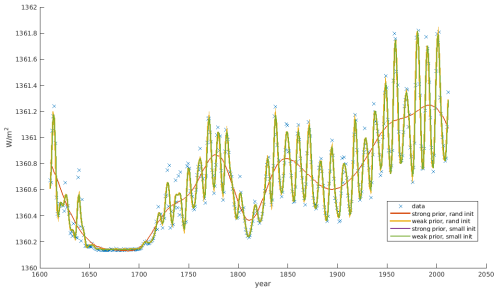


Figure 3: GP predictions on solar data set with SE kernel for different priors and initialisations.

Appendix G. Optimisation

Since we wish to use the learned kernels for interpretation, it is important to have the hyperparameters lie in a sensible region after the optimisation. In other words, we wish to regularise the hyperparameters during optimisation. For example, we want the SE kernel to learn a globally smooth function with local variation. When naïvely optimising the lower bound, sometimes the length scale and the signal variance becomes very small, so the SE kernel explains all the variation in the signal and ends up connecting the dots. We wish to avoid this type of behaviour. This can be achieved by giving priors to hyperparameters and optimising the energy (log prior added to the log marginal likelihood) instead, as well as using sensible initialisations. Looking at Figure 3, we see that using a strong prior with a sensible random initialisation (see Appendix H for details) gives a sensible smoothly varying function, whereas for all the three other cases, we have the length scale and signal variance shrinking to small values, causing the GP to overfit to the data.

Careful initialisation of hyperparameters and inducing points is also very important, and can have strong influence the resulting optima. It is sensible to have the optimised hyperparameters of the parent kernel in the search tree be inherited and used to initialise the hyperparameters of the child. The new hyperparameters of the child must be initialised with random restarts, where the variance is small enough to ensure that they lie in a sensible region, but large enough to explore a good portion of this region. As for the inducing points, we want to spread them out to capture both local and global structure. Trying both K-means and a random subset of training data, we conclude that they give similar results and resort to a random subset. Moreover we also have the option of learning the inducing points. However, this will be considerably more costly and show little improvement over fixing them, as we show in Section 4. Hence we do not learn the inducing points, but fix them to a given set.

Appendix H. Hyperparameter initialisation and priors

$Z \sim \mathcal{N}(0,1), TN(\sigma^2, I)$ is a Gaussian with mean 0 and variance σ^2 truncated at the interval I then renormalised.

Signal noise

$\sigma^2 = 0.1 \times \exp(Z/2)$

$p(\log \sigma^2) = \mathcal{N}(0, 0.2)$

LIN
$\sigma^2 = \exp(V)$ where $V \sim TN(1, [-\infty, 0])$
$p(\log \sigma^2) = logunif$

SE
$l = \exp(Z/2), \sigma^2 = 0.1 \times \exp(Z/2)$
$p(\log l) = \mathcal{N}(0, 0.01), p(\log \sigma^2) = logunif$

PER
$p_{min} = \log(10 \times \frac{\max(x) - \min(x)}{N})$ (shortest possible period is 10 time steps)
$l = \exp(Z/2), p = \exp(p_{min} + W), \sigma^2 = 0.1 \times \exp(Z/2)$ where $W \sim \mathcal{TN}(-0.5, [0, \infty))$
$p(\log l) = t(\mu = 0, \sigma^2 = 1, \nu = 4), p(\log p) = \mathcal{LN}(p_m in - 0.5, 1), p(\log \sigma^2) = logunif$ where $\mathcal{LN}(\mu, \sigma^2)$ is log Gaussian, $t(\mu, \sigma^2, \nu)$ is the student's t-distribution.

Appendix I. Computation Times for CKS

Table 2: Mean and standard deviation of computation times (in seconds) for full GP optimisation, Var GP optimisation, NLD and NIP (PCG using PIC preconditioner) upper bounds over 10 random iterations.

		Solar	**Mauna**	**Concrete**
GP		29.1950 ± 5.1430	164.8828 ± 58.7865	403.8233 ± 127.0364
Var GP,	m=10	7.0259 ± 4.3928	6.0117 ± 3.8267	5.4358 ± 0.7298
	m=20	8.3121 ± 5.4763	11.9245 ± 6.8790	10.2410 ± 2.5109
	m=40	10.2263 ± 4.1025	17.1479 ± 10.7898	19.6678 ± 4.3924
	m=80	9.6752 ± 6.5343	28.9876 ± 13.0031	47.2225 ± 13.1955
	m=160	25.6330 ± 8.7934	91.0406 ± 39.8409	158.9199 ± 18.1276
	m=320	76.3447 ± 20.3337	202.2369 ± 96.0749	541.4835 ± 99.6145
NLD,	m=10	0.0019 ± 0.0001	0.0033 ± 0.0002	0.0113 ± 0.0004
	m=20	0.0026 ± 0.0001	0.0046 ± 0.0002	0.0166 ± 0.0007
	m=40	0.0043 ± 0.0001	0.0079 ± 0.0003	0.0286 ± 0.0005
	m=80	0.0084 ± 0.0002	0.0154 ± 0.0004	0.0554 ± 0.0012
	m=160	0.0188 ± 0.0006	0.0338 ± 0.0007	0.1188 ± 0.0030
	m=320	0.0464 ± 0.0032	0.0789 ± 0.0036	0.2550 ± 0.0074
NIP,	m=10	0.0474 ± 0.0092	0.1020 ± 0.0296	0.2342 ± 0.0206
	m=20	0.0422 ± 0.0130	0.1274 ± 0.0674	0.1746 ± 0.0450
	m=40	0.0284 ± 0.0075	0.0846 ± 0.0430	0.2345 ± 0.0483
	m=80	0.0199 ± 0.0081	0.0553 ± 0.0250	0.2176 ± 0.0376
	m=160	0.0206 ± 0.0053	0.0432 ± 0.0109	0.2136 ± 0.0422
	m=320	0.0250 ± 0.0019	0.0676 ± 0.0668	0.2295 ± 0.0433
Var GP,	m=10	23.4 ± 14.6	42.0 ± 33.0	110.0 ± 302.5
learn IP	m=20	38.5 ± 17.5	62.0 ± 66.0	70.0 ± 97.0
	m=40	124.7 ± 99.0	320.0 ± 236.0	307.0 ± 341.4
	m=80	268.6 ± 196.6	1935.0 ± 1103.0	666.0 ± 41.0
	m=160	1483.6 ± 773.8	10480.0 ± 5991.0	4786.0 ± 406.9
	m=320	2923.8 ± 1573.5	39789.0 ± 23870.0	25906.0 ± 820.9

Appendix J. Analytic upper bound to marginal likelihood

If we can find an analytic upper bound to the marginal likelihood, whose value and gradients can be evaluated in linear time, then we can maximise this to get an upper bound of exact likelihood with optimal hyperparameters. Hence for any m, we can find an interval that contains the true optimised marginal likelihood. So if this interval is dominated by an interval of another kernel, we can discard the kernel and there is no need to evaluate the bounds for bigger values of m. Now we wish to use values of m such that we can choose the right kernel (or kernels) at each depth of the search tree with minimal computation. This gives rise to an exploitation-exploration trade-off, whereby we want to balance between raising m for tight intervals that allow us to discard unsuitable kernels whose intervals fall strictly below that of other kernels, and quickly moving on to the next depth in the search tree to search for finer structure in the data. The search algorithm is highly parallelisable, and thus we may raise m simultaneously for all candidate kernels. At deeper levels of the search tree, there may be too many candidates for simultaneous computation, in which case we may select the ones with the highest upper bound to get tighter intervals. Such attempts to find an analytic upper bound are shown below.

First note that

$$-\frac{1}{2}\log\det(\hat{K}+\sigma^2 I)-\frac{1}{2}y^\top(K+\sigma^2 I)^{-1}y$$

is an upper bound to the marginal likelihood, up to a constant. Moreover from Equation (5), we have that

$$-\frac{1}{2}\log\det(\hat{K}+\sigma^2 I)+\frac{1}{2}\alpha^\top(K+\sigma^2 I)\alpha-\alpha^\top y$$

is an upper bound $\forall\alpha\in\mathbb{R}^N$. Thus one idea of obtaining a cheap upper bound to the optimised marginal likelihood was to solve the following maximin optimisation problem:

$$\max_\theta \min_{\alpha\in\mathbb{R}^N} -\frac{1}{2}\log\det(\hat{K}+\sigma^2 I)+\frac{1}{2}\alpha^\top(K+\sigma^2 I)\alpha-\alpha^\top y$$

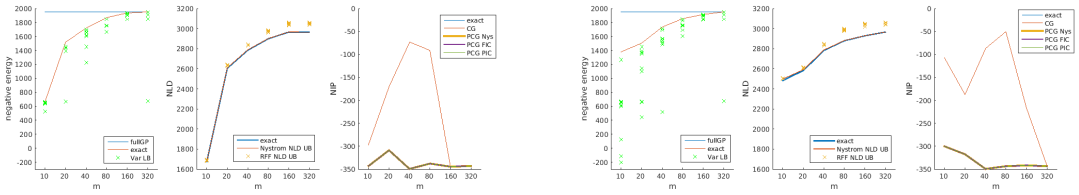
One way to solve this cheaply would be by coordinate descent, where one maximises with respect to θ fixing α, then minimises with respect to α fixing θ. However σ tends to blow up in practice. This is because the expression is $O(-\log\sigma^2+\sigma^2)$ for fixed α, hence maximising with respect to σ pushes it towards infinity.

An alternative is to use the approximate upper bound

$$-\frac{1}{2}\log\det(\hat{K}+\sigma^2 I)-\frac{1}{2}y^\top(\hat{K}_{PIC}+\sigma^2 I)^{-1}y$$

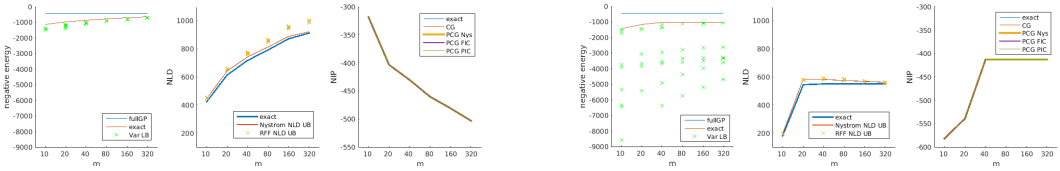
However we see that maximising this bound gives quite a loose upper bound unless $m=O(N)$. Hence this upper bound is not very useful.

Appendix K. Further Plots



(a) Mauna: fix inducing points (b) Mauna: learn inducing points

Figure 4: Same as Figures 1(a) and 1(b) but for Mauna Loa data.



(a) Concrete: fix inducing points (b) Concrete: learn inducing points

Figure 5: Same as Figures 1(a) and 1(b) but for Concrete data.

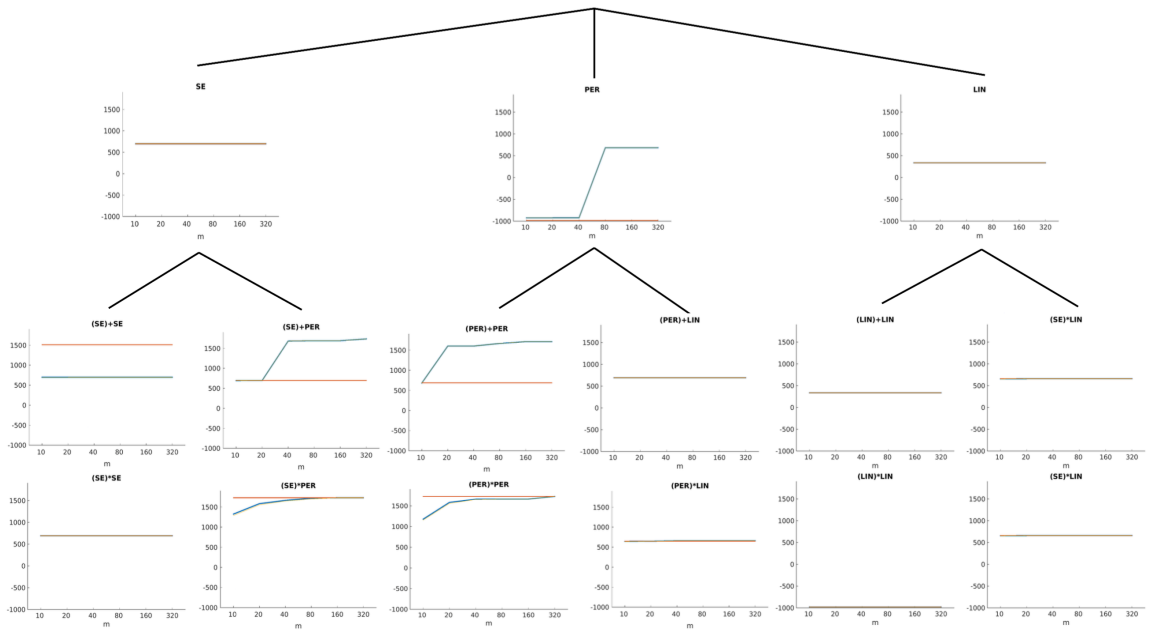


Figure 6: Same as Figure 2 but for Mauna data.

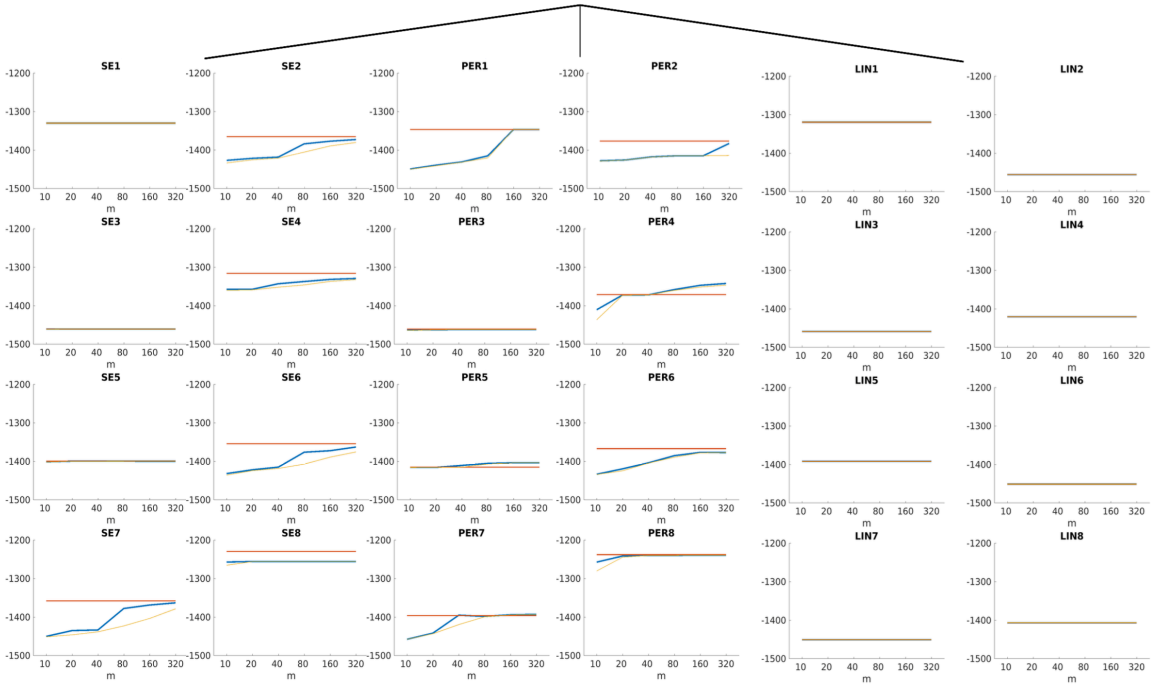


Figure 7: Same as Figure 2 but for concrete data and up to depth 1.